

AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated in the following listing of all claims:

1. (Currently amended) A method of profiling code for an execution environment in which latency exists between an execution event and detection thereof, the method comprising:
 - executing the code on a processor, the code including, for at least one operation, a skid region between the at least one operation and a subsequent ambiguity creating location;
 - detecting the execution event, wherein the skid region provides for detection of the execution event prior to encountering the subsequent ambiguity creating location;
 - and
 - backtracking from a detection point in the code coinciding with the detection of the execution event to a preceding operation associated with the execution event, the backtracking identifying the preceding operation at a ~~predetermined displacement from the detection point unless an ambiguity creating location is disposed therebetween, wherein the predetermined displacement is dependent upon the preceding operation which is indicated by the execution event.~~

2. (Original) The method of claim 1,
 - wherein the ambiguity creating location is a branch target location.

3. (Currently amended) The method of claim 2,
 - wherein the ~~ambiguity creating location is bridged using branch history information~~ preceding operation corresponds to a load instruction.

4. (Original) The method of claim 1,
 - wherein the ambiguity creating location is an entry point location.

5. (Original) The method of claim 1, wherein the ambiguity creating location is one of:
a jump target location;
an indirect branch target location;
a trap handler location; and
an interrupt handler location.
6. (Original) The method of claim 1,
wherein the preceding operation corresponds to a load instruction; and
wherein the execution event is a cache miss.
7. (Original) The method of claim 1,
wherein the preceding operation corresponds to a memory access instruction; and
wherein the execution event is either a hit or a miss at a level in a memory hierarchy.
8. (Previously Presented) The method of claim 1,
wherein the execution event is either an overflow or an underflow of a hardware counter.
9. (Previously Presented) The method of claim 1,
wherein the execution event triggers either an overflow or an underflow of a hardware
counter that is itself detected.
10. (Original) The method of claim 1,
wherein the latency includes that associated with delivery of a trap.
11. (Original) The method of claim 1,
wherein the latency includes that associated with delivery of a counter overflow event
signal.
12. (Original) The method of claim 1,
wherein the latency is associated with pipeline execution skid.

13. (Original) The method of claim 1,
wherein the latency is associated with completion of in-flight operations.
14. (Original) The method of claim 1, embodied in a computer program product.
15. (Original) The method of claim 1, embodied in at least one of:
a profiling tool;
a code optimizer; and
a runtime library.
16. (Currently amended) The method of claim 1, employed in combination with a compiler that pads the code, to provide the skid region, with one or more padding operations to absorb at least some instances of the latency.
17. (Original) The method of claim 16,
wherein the padding operations are not themselves associated with the execution event.
18. (Original) The method of claim 16,
wherein the padding operations are not themselves ambiguity creating locations.
19. (Currently amended) A method of identifying operations associated with execution events of a processor, the method comprising:
providing a skid region between a target of a control transfer and a particular operation;
from a point in an execution sequence of the operations of the processor, the point coinciding with an execution event, backtracking through the operations toward
[[a]] the particular operation that precedes the coinciding point by a
predetermined latency; and
associating the execution event with the particular operation, wherein the skid region
provides for detection of the execution event prior to encountering the target of
the control transfer~~unless the backtracking encounters an unresolved intervening~~
~~target of a control transfer, wherein the predetermined latency is dependent upon~~
~~the particular operation which is indicated by the execution event.~~

20. (Previously Presented) The method of claim 19, further comprising:
executing the sequence of operations on the processor; and
detecting the execution event.
21. (Previously Presented) The method of claim 19,
wherein the operations are instructions executable on the processor; and
wherein the particular operation is a particular one of the instructions that triggers the
execution event.
22. (Previously Presented) The method of claim 19,
wherein the operations correspond to instructions of program code.
23. (Original) The method of claim 19,
wherein the execution event is an exception triggering execution of the particular
operation.
24. (Original) The method of claim 19,
wherein the execution event is a cache miss.
25. (Currently amended) The method of claim 19,
wherein the ~~expected~~ latency includes a trap delivery delay.
26. (Currently amended) The method of claim 19,
wherein the execution event triggers a hardware event and the ~~expected~~ latency includes
delivery of a signal associated therewith.
27. (Original) The method of claim 26,
wherein the hardware event is either underflow or overflow of a counter associated with
the execution event.
28. (Original) The method of claim 19,
wherein the execution event is either underflow or overflow of a counter.

29. (Currently amended) The method of claim 19,
wherein instances of ~~intervening~~ control transfer targets are identified in the execution
sequence of operations to facilitate the backtracking.
30. (Currently amended) The method of claim 29,
wherein ~~at least some of the instances of intervening control transfer targets are resolved~~
using branch history information the skid region is added by a compiler to avoid
encountering intervening ones of the control transfer targets during the
backtracking.
31. (Original) The method of claim 19,
wherein control transfer target locations in the execution sequence are identified by a
compiler.
32. (Previously Presented) The method of claim 19,
wherein the operations are instructions executable on the processor; and
wherein the particular operation is a particular one of the instructions that triggers the
execution event.
33. (Previously Presented) The method of claim 19,
wherein the operations correspond to instructions of program code.
34. (Original) The method of claim 19, further comprising:
preparing the execution sequence of the operations.
35. (Original) The method of claim 34,
wherein preparation of the execution sequence includes identifying a location of the
control transfer target therein.
36. (Original) The method of claim 34,
wherein preparation of the execution sequence includes identifying a location of the
particular operation therein.

37. (Original) The method of claim 19,
wherein the particular operations include memory referencing instructions.

38. (Original) The method of claim 37,
wherein the memory referencing instructions include one or more of loads, stores and
prefetches.

39. (Currently amended) A method of associating an execution characteristic of code
executed on a processor with a particular operation thereof, the method comprising:
identifying at least first-type and second-type operations in the code;
providing a skid region between an operation of the first-type and an operation of the
second-type; and
from a point in an execution sequence of the code that coincides with delayed detection
of the execution characteristic, backtracking a ~~predetermined~~ displacement toward
a candidate triggering operation of the first-type and associating the candidate
triggering operation with the execution characteristic, wherein the skid region
provides for detection of the execution characteristic prior to encountering the
operation of the second-type.~~unless an unresolved intervening operation of the~~
~~second type is encountered, wherein the predetermined displacement is dependent~~
~~upon the first-type operations which are indicated by the execution characteristic.~~

40. (Original) The method of claim 39,
wherein the first-type operations include memory access operations.

41. (Original) The method of claim 39,
wherein the second-type operations include operations that coincide with control transfer
target locations in the code.

42. (Original) The method of claim 39,
wherein the execution characteristic involves memory access latency.

43. (Original) The method of claim 39,

wherein the execution characteristic includes a cache miss statistic.

44. (Original) The method of claim 39,
wherein the detection delay includes a pipelined execution skid latency.

45. (Currently amended) The method of claim 39,
wherein the second-type operations include operations that coincide with branch target
locations in the code; and
wherein at least some of the branch target locations are ~~resolved using branch history~~
information avoided during the backtracking.

46. (Currently amended) A method of preparing code for a processor, the method
comprising:
preparing a tangible first executable instance of the code, the preparing identifying at
least ambiguity creating locations therein;
providing a skid region between a particular operation and a subsequent one of the
ambiguity creating locations;
executing the first executable instance and responsive to detection of an execution
characteristic, backtracking a ~~predetermined~~ displacement through the code to
identify an associated operation thereof, wherein the skid region provides for
detection of the execution characteristic associated with the particular operation
prior to encountering the subsequent one of the ambiguity creating locations;
~~extent of the backtracking is limited at least by encountering of an unresolved~~
~~intervening one of the identified ambiguity creating locations, wherein the~~
~~predetermined displacement is dependent upon the associated operation which is~~
~~indicated by the execution characteristic;~~ and
further preparing a tangible second executable instance of the code using the association
between the associated operation and the execution characteristic.

47. (Previously Presented) The method of claim 46,
wherein the association between the associated operation and the execution characteristic
is based on a set of additional detections and responsive backtracking.

48. (Original) The method of claim 46,
wherein the execution characteristic involves memory access latency; and
wherein the preparation of the second executable instance includes insertion of prefetch
operations into the code.

49. (Currently amended) The method of claim 46, further comprising:
~~resolving at least some intervening ones of~~ avoiding at least some of the identified
ambiguity creating locations ~~using branch history information during the~~
backtracking.

50. (Currently amended) A computer program product encoded in one or more
computer readable media, the computer program product comprising:
an execution sequence of operations; and
padding operations following at least some particular operations of the execution
sequence, the padding operations providing an unambiguous skid region of the
execution sequence between the at least some particular operations and
subsequent ambiguity creating locations to provide for detection of execution
events prior to encountering the subsequent ambiguity creating locations and a
~~predetermined latency between the ambiguity creating locations and the at least~~
~~some particular operations to facilitate~~ determination of which of the at least some
particular operations are associated with the execution events.

51. (Original) The computer program product of claim 50,
wherein the particular operations include memory access operations.

52. (Original) The computer program product of claim 50,
wherein the padding operations include nops.

53. (Original) The computer program product of claim 50,
wherein the unambiguous skid region does not include an ambiguity creating location.

54. (Previously Presented) The computer program product of claim 50,

wherein the one or more computer readable media are selected from the set of a disk, tape, magnetic, optical, semiconductor or electronic storage medium and a network, wireline, or wireless communications medium.

55. (Currently amended) A computer program product encoded in one or more computer readable media, the computer program product comprising:
 an execution sequence of operations; and
 one or more data sections that identify in the execution sequence at least ambiguity creating locations and target operations for use by one or both of a profiler and a optimizer, wherein the target operations occur at predetermined displacements from execution events and a skid region is provided between at least one of the target operations and a respective one of the ambiguity creating locations to provide for detection of one of the execution events prior to encountering the respective one of the ambiguity creating locations. ~~unless at least one of the at least ambiguity creating locations is disposed therebetween, and wherein the predetermined displacements are dependent upon the target operations which are indication by the execution events.~~

56. (Original) The computer program product of claim 55,
 wherein the ambiguity creating locations include branch target locations.

57. (Original) The computer program product of claim 55,
 wherein the target operations include memory referencing operations.

58. (Previously Presented) The computer program product of claim 55,
 wherein the one or more computer readable media are selected from the set of a disk, tape, magnetic, optical, semiconductor or electronic storage medium and a network, wireline, or wireless communications medium.

59. (Currently amended) An apparatus comprising:
 means for backtracking a predetermined placement, from a point coinciding with an execution event in an execution sequence of operations on a processor, through

the execution sequence toward a particular operation thereof that precedes the coinciding point; and

means for associating the execution event with the particular operation by providing a skid region between the particular operation and a subsequent ambiguity creating location, wherein the skid region provides for detection of the execution event prior to encountering the subsequent ambiguity creating location. ~~unless the backtracking encounters an intervening ambiguity creating location, wherein the predetermined displacement is dependent upon the particular operation which is indicated by the execution event.~~

60. (Currently amended) The apparatus of claim 59, ~~further comprising:~~ wherein the particular operation is a memory access operation.

~~means for bridging at least some ambiguity creating locations.~~

61. (Currently amended) An apparatus comprising:

a code preparation facility suitable for preparation of an execution sequence of operations for a processor; and

means for padding the execution sequence to provide an unambiguous skid region between a particular operation and a subsequent ambiguity creating location within the sequence of operations to provide for detection of the execution event prior to encountering the subsequent ambiguity creating location. ~~a predetermined latency between the ambiguity creating location and the particular operation to facilitate determination of which of the sequence of operations is associated with an execution event.~~